Title of Grant / Cooperative Agreement:	A Framework for Turbulence Modeling Using Big Data
Type of Report:	Summary of Research
Name of Principal Investigator:	Karthikeyan Duraisamy
Period Covered by Report:	01/01/2014-12/31/2015
Name and Address of recipient's institution:	Regents of the University of Michigan, 503 Thompson ST, Ann Arbor MI 48109
NASA Grant / Cooperative Agreement Number:	NNX14AC72A

Reference 14 CFR § 1260.28 Patent Rights *(abbreviated below)*
The Recipient shall include a list of any Subject Inventions required to be disclosed during the preceding year in the performance report, technical report, or renewal proposal. A complete list (or a negative statement) for the entire award period shall be included in the summary of research.

Subject inventions include any new process, machine, manufacture, or composition of matter, including software, and improvements to, or new applications of, existing processes, machines, manufactures, and compositions of matter, including software.

Have any Subject Inventions / New Technology Items resulted from work performed under this Grant / Cooperative Agreement?	No ◉	Yes ○
If yes a complete listing should be provided here: Details can be provided in the body of the Summary of Research report.		

Reference 14 CFR § 1260.27 Equipment and Other Property (abbreviated below)
A Final Inventory Report of Federally Owned Property, including equipment where title was taken by the Government, will be submitted by the Recipient no later than 60 days after the expiration date of the grant. Negative responses for Final Inventory Reports are required.

Is there any Federally Owned Property, either Government Furnished or Grantee Acquired, in the custody of the Recipient?	No ◉	Yes ○
If yes please attach a complete listing including information as set forth at § 1260.134(f)(1).		

Attach the Summary of Research text behind this cover sheet.

Reference 14 CFR § 1260.22 Technical publications and reports (December 2003)
Reports shall be in the English language, informal in nature, and ordinarily not exceed three pages (not counting bibliographies, abstracts, and lists of other media).

A Summary of Research (or Educational Activity Report in the case of Education Grants) is due within 90 days after the expiration date of the grant, regardless of whether or not support is continued under another grant. This report shall be a comprehensive summary of significant accomplishments during the duration of the grant.

A Framework for Turbulence Modeling using Big Data: Phase I Report

Karthik Duraisamy (PI)
University of Michigan, Ann Arbor, MI

Juan J. Alonso (Co-I)
Stanford University, Stanford, CA

Paul Durbin (Co-I)
Iowa State University, Ames, IA

Contents

1 Introduction

Accurate simulation of transitional and turbulent flows is of critical importance to many applications in science and engineering. The onset of turbulence is considered undesirable in many situations such as those in turbo-machinery, atmospheric re-entry vehicles, commercial aircraft, *etc.* due to consequences such as increased losses, aerodynamic heating, and decreased fuel efficiency. In contrast, turbulence plays a beneficial role in settings such as mixing and combustion of reacting gases. Despite the tremendous growth in computational resources over the past decade, modeling and simulation of many practical/realistic turbulent flows – to the desired level of accuracy – has remained challenging, and in some cases, even inaccessible. Though Direct Numerical Simulations (DNS) and Large Eddy Simulations (LES) have offered tremendous insight and predictive capabilities in many flows, these techniques continue to be infeasible for high Reynolds number wall-bounded flows. This situation is unlikely to change within the next few decades unless significant advances are made in hybrid techniques that employ a near-wall model in conjunction with an outer-layer LES. Thus, near-wall models in both an LES and in a Reynolds-averaged Navier–Stokes (RANS) context will be the pacing item in applied computational fluid dynamics (CFD). This assertion is also supported by the findings of a recent NASA study[28].

Popular turbulence (and transition) models are all drastic simplifications to the rich dynamics of turbulence. Turbulence closure equations usually introduce between one and seven additional transport variables and many adjustable constants selected by the engineering judgment of the modeler. These constants are calibrated by a small number of simple test cases such as homogeneous turbulence and thin-shear flows, and given this development process, it is unsurprising that accuracy diminishes as the model is applied to problems which deviate from the calibrated cases. Typical examples in which RANS models are deficient involve adverse pressure gradients, inhomogeneous flow directions, secondary effects and flow separation. Practical turbulence models have been simple out of expediency. However, in attempts to increase accuracy, some work has shifted toward greater complication in turbulence models. As an example, Refs. 1, 2 introduce additional tensorial bases to account for inhomogeneity, wall echo, and anisotropies in second moment closures. While these efforts are encouraging, the potential benefits are obscured by the need to determine a number of free parameters from a small set of often idealized test cases. New strategies are required to move beyond these limitations and we believe that data-driven approaches are capable of providing solutions.

Data science is on the rise in many disciplines due to improvements in computational power and the increased availability of large data sets. This has been accompanied by significant improvements in data analytics and machine learning (ML) techniques, both in effectiveness and scalability. Various ML techniques are widely used today in financial and commercial applications such as stock trading, fraud detection, preference choices, etc. and scientific applications such as genomics, astrophysics, fluid mechanics, and natural language processing. Depending on the application, the objectives of the tasks can be a combination of automated clustering and classification, feature extraction, predictive modeling and improved decision making. Specifically in the area of turbulent flows, previous efforts have used neural networks for near-wall modeling through reconstruction of structures in a fully-developed turbulent channel flow[3], real-time extraction of coherent spatio-temporal structures[4], and optimization of closure coefficients of the two-equation k-ϵ turbulence model [5], *etc.* Other attempts (Refs. 30,31) have approached the problem from the viewpoint of structural uncertainty quantification or Bayesian model averaging[32, 33].

Very recently, data-driven statistical inference to correct for model error (Edeling et al. [6]) has been proposed to address some of the deficiencies of a priori processing. This, and other approaches have focused on estimating the parameters of the standard models to calibrate a set of pre-specified building-block functions. The uniqueness of our approach stems from the focus on inferring and reconstructing deficiencies in the **functional form** of known turbulence models, rather than on the model parameters. Further, in contrast to data-driven *descriptive* modeling approaches, data-driven *predictive* modeling involves additional challenges. The enabling tools in our approach are inverse modeling and machine learning.

In Phase I of this LEARN project, we develop the formalism and tools to infuse closure models of transition, turbulence and other fields of mechanics with data-driven aspects. First, a proof-of-concept study is performed in which a known turbulence model is assumed to for the surrogate truth and a Machine Learning technique is assessed in its ability to replace key terms in the turbulence model. Following this exercise, the key steps of inversion, learning and injection are exercised in turbulence and transition problems.

2 Machine Learning: Proof-of-concept

In this section, a proof-of-concept of the data-driven approach of turbulence model development is presented. Specifically, we investigate the feasibility of this approach by attempting to reproduce, through a machine learning methodology, the results obtained with the well-established Spalart-Allmaras model. In other words, the key question that we seek to answer is the following: Given a number of observations of CFD solutions using the Spalart-Allmaras model (our *truth* model), can we reproduce those solutions using machine-learning techniques *without knowledge of the structure, functional form, and coefficients of the actual model?*

2.1 Machine Learning

Feed-forward neural networks[40] are a balance between parametric and non-parametric learning algorithms. A feed-forward neural network is a directed acyclic graph consisting of an input layer, some number of hidden layers, and an output layer. The net begins with an input feature vector (x_i) which is used as the input to the first hidden layer. The outputs from each layer in turn are "fed forward" as inputs to the next layer, finishing with the final layer whose output is taken as the prediction for the quantities of interest. Each hidden layer and the output layer are comprised of a set of neurons. A neuron is a simple computational unit that takes a weighted sum of its inputs (which are the inputs to the layer), sends that weighted sum through an activation function, and outputs the final result.

More specifically, assume the neural network has L layers, i.e. $L-1$ hidden layers and one output layer. Let K_L be the number of neurons in layer L, and let $Z_{l,k}$ be the output of the k^{th} neuron in layer l. The inputs to all of the neurons in layer l are the ouputs from layer $l-1$. Each neuron in layer l has $K_{l-1}+1$ weights, $w_{l,k,0}, w_{l,k,1}, ..., w_{l,k,K_{l-1}}$, with one weight per input to the neuron and one additional weight acting as a bias term. The output of each neuron is computed by taking a weighted sum of the inputs and then computing the neuron's activation function, $g_{l,k}$, of that weighted sum as follows:

$$Z_{l,k} = g_{l,k}\left(w_0 + \sum_{i=1}^{K_{l-1}} w_{l,k,i} Z_{l-1,i}\right) \quad . \tag{1}$$

The initial set of outputs, $Z_{0,k}$ are simply the inputs to the neural network. The number of hidden layers and the number of neurons per layer are free parameters chosen by the user of the neural network. Activation functions for the neurons in the hidden layers are almost always selected to be monotonic, non-linear, and to have finite asymptotic values, and typically only one type of activation function is used. The activation functions for the neurons in the output layer are usually chosen as the identity function in regression problems, i.e., $g(x) = x$.

The weight variables $w_{i,j,k}$ are free parameters of the neural net. Typically, the user chooses a "loss function", which is a function of the predicted value and the true value at an input. The weights of the neural net are set using some form of numerical optimization in order to minimize the loss function (in our case the gradient-based algorithms mentioned briefly earlier):

$$\text{minimize}_{w_{i,j,k}} \sum_i l\left(p(x_i; w_{i,j,k}), t_{x_i}\right) \tag{2}$$

where $p(x_i; w_{i,j,k})$ is the value at data point x_i predicted by the neural net for the given choice of weights, t_{x_i} is the true value at x_i in the data set, and $l(.)$ is the *loss function*.

Strictly, a neural network is a parametric fit to the data. The functional form is the non-linear composition of activator functions described above, and the parameters are the weights of the neurons. However, this composition makes neural networks particularly flexible in the range of functions they can accurately represent, and the number of free parameters is easily increased. In fact, it can be shown [41] that a sufficiently large two-layer neural network can approximate any function to high accuracy. As a result, the number of hidden layers is typically kept small < 4, while the number of neurons per layer is scaled as accuracy demands.

Neural networks seem particularly attractive for turbulence modeling. They are flexible in the range of functions they can fit, but once the training is complete the functional form is a compact set of equations represented by a small number of parameters. The neural network prediction is continuous and smooth (assuming the activator functions are chosen as such), and the computation time for a datapoint is cheap. For a fixed network size, the neural network prediction time is unrelated to the size of the training data allowing the use of very large datasets, and gradients of the prediction error with respect to the neuron weights can be efficiently computed allowing the use of gradient-based optimization to find the optimal set of weights.

2.2 Problem Setup

The Spalart-Allmaras turbulence model is a one-equation closure to the RANS equations that models the transport of turbulent kinematic viscosity, $\hat{\nu}$. The model constructs the Reynolds-stress tensor, $\tau_{i,j}$, by use of the Boussinesq approximation, and computing the turbulent eddy viscosity, μ_T, by

$$\mu_T = \rho\hat{\nu} f_{v1} \quad ; \; f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \quad ; \; \chi = \frac{\hat{\nu}}{\nu} \quad . \tag{3}$$

The model assumes $\hat{\nu}$ is convected and diffused with the flow, and defines the production and destruction of $\hat{\nu}$ using the wall distance and local flow quantities. The full model is given as:

$$\frac{\partial \hat{\nu}}{\partial t} + u_j \frac{\partial \hat{\nu}}{\partial x_j} = c_{b1}(1 - f_{t2})\hat{S}\hat{\nu} - \left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2} \right) \left(\frac{\hat{\nu}}{d} \right)^2 + \frac{1}{\sigma} \left(\frac{\partial}{\partial x_j} \left((\nu + \hat{\nu}) \frac{\partial \hat{\nu}}{\partial x_j} \right) + c_{b2} \frac{\partial \hat{\nu}}{\partial x_i} \frac{\partial \hat{\nu}}{\partial x_i} \right) \quad . \tag{4}$$

The convective and diffusive terms have natural definitions, but the correct definition for the source term is less clear. The above source definition was constructed using the knowledge/experience of the model authors and simplified flow conditions to tune the values of the unknown constants. While the SA model gives reasonably-good predictions for many attached flows of interest, it is probable that a better formulation of the source term could result in better predictions over a wider range of flows. The question is how to discover such a formulation of the terms that have been explicitly modeled in the S-A equation.

Machine learning provides a novel way to aid in the construction of more accurate turbulence models by relying on a suite of high-fidelity datasets in order to construct a more accurate closure term formulation, instead of simply using the modeler's experience, knowledge, and intuition, and a small set of calibration flow fields. For example, a well-resolved DNS simulation provides a field of both mean-flow quantities such as velocity and pressure gradients, and turbulent quantities such as the Reynolds stress tensor at every grid location. Each grid location in this field contains a piece of information about the evolution of turbulence as a function of mean quantities and their gradients. A suite of DNS computations, augmented with data from LES and experimental data, can therefore provide a large corpus of data of the relationship between mean-flow and turbulent quantities. It follows that a turbulence modeler can use machine learning to pre-process this large quantity of data, for example to enhance the source term of the SA model. A modeler would project the DNS Reynolds stress tensor into a turbulent eddy viscosity [42] pointwise, giving a field of $\hat{\nu}$.

The source could then be computed from the difference of the convection and diffusion. A modeler then chooses a set of input features (vorticity, viscosity ratio, etc.), and trains a machine learning algorithm to find the source of $\hat{\nu}$ as a function of the chosen inputs. This new "equation" for the source term (the input-output relationship learned from by the machine learning algorithm) can be used inside the CFD solver instead of the original equation. Note that no assumptions have been made about the functional form of the relationship; an entirely new functional mapping is created instead of merely enhancing an existing model.

Given a good choice of input features, a varied set of training cases, and a correct training of the machine-learning algorithm, this new source term should, in theory, perform better than the original. Of course, this is easier said than done, and while the idea is very promising, there are many basic questions to be answered. Can machine learning be successful on CFD data? Will such an algorithm provide similar stability to a hand-crafted model? Will the learned model successfully predict the result on unseen flows? How much data is sufficient? Is more data always better? Must we pre-process / pre-select the data prior to providing it to the machine learning algorithm?

It is difficult to answer these questions by beginning with DNS data to inform the closure terms of existing RANS models. With DNS data, a complete and compact set of input features is unknown, and even the best mapping of features is likely to contain significant amounts of noise. In such an environment, it is hard to distinguish fundamentals flaws in the learning process from the difficulties in dealing with real data. Instead, it is instructive to begin in a controlled environment in which properties of the learning process can be explored in isolation.

2.2.1 Model problems

We will take case of the Spalart-Allmaras turbulence modeal as a model problem, and use machine-learning to attemps to reproduce its behavior. The SA source term is a known, analytic function and thus the correct inputs and the outputs are avaiable. For exploratory purposes, we can assume that Spalart-Allmaras represents the true behavior of turbulence, and see if a machine-learned model of SA correctly reproduces its results. In this sandbox, the correct answer is known, and any differences between the ML and SA flow solutions can be attributed to difficulties with the learning process (as opposed to noisy data, incorrect input choices, etc.). Additionally, such a study firmly establishes a methodology for future ML experiments using DNS (or LES) simulation results.

There are many ways to "learn" the SA source term. The full source term, s, could be replaced, or it could broken into several parts: the production

$$s_p = c_{b1}(1 - f_{t2})\hat{S}\hat{\nu} \ , \tag{5}$$

destruction

$$s_d = \left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2} \right) \left(\frac{\hat{\nu}}{d} \right)^2 \ , \tag{6}$$

and/or cross production

$$s_{cp} = \frac{c_{b2}}{\sigma} \frac{\partial \hat{\nu}}{\partial x_i} \frac{\partial \hat{\nu}}{\partial x_i} \ . \tag{7}$$

The production could be further simplified to studying only the multiplier of $\hat{S}\hat{\nu}$, $m_p = c_{b1}(1 - f_{t2})$, and the destruction term could be simplified to only consider the multiplier to $(\hat{\nu}/d)^2$, $m_d = c_{w1} f_w - (c_{b1}/\kappa^2) f_{t2}$. The learned model could be used in certain regions of the flow, for example exclusively within the boundary layer, or applied in the entire flow domain.

For any of these choices, the basic procedure we follow is the same:

1. Collect a portfolio of flow solutions of interest by running the *true* S-A model in a suitable CFD solver.

2. From each grid location in each flow solution, extract and/or compute an input feature vector and an output feature vector.

3. Construct the training set by concatenating the feature vectors from step 2.

4. Choose an appropriate machine learning algorithm and loss function.

5. Train the ML algorithm on the training set from 3. This creates a functional mapping between the chosen inputs and outputs (i.e. constructs a model of the source term).

6. Integrate this learned model into the CFD solver, and run a representative set of validation cases to test the behavior of the machine-learned source term.

We discuss in greater detail the choices for the input feature vector and loss functions below.

2.2.2 Local Non-dimensionalization

The source term is known to be a function of five local flow quantities, ν, $\hat{\nu}$, Ω, d, and $N = \frac{\partial\hat{\nu}}{\partial x_i}\frac{\partial\hat{\nu}}{\partial x_i}$. These quantities, however, are not an appropriate choice for the input feature vector to the machine learning algorithm. They are dimensional quantities which may have different numeric values even when two flows are dynamically similar. While an ML algorithm could be trained on these dimensional quantities, there is nothing to gain by doing so; transforming the inputs such that all flows have the same scale makes the data more compact, loses no generality, and helps prevent overfitting by the ML algorithm. We rescale the inputs using $\nu = \nu^*$ and $L = L^*$ (where L is the Reynolds length scale). The flow quantities used in the ML process then become

$$\hat{\nu}^* = \hat{\nu}/\nu^* \ , \tag{8}$$

$$d^* = d/L^* \ , \tag{9}$$

$$\Omega^* = \frac{L^{*2}}{\nu^*}\Omega \ , \tag{10}$$

$$N^* = \frac{L^{*2}}{\nu^{*2}}N \ , \tag{11}$$

$$s_i{}^* = \frac{L^{*2}}{\nu^{*2}}s_i \ . \tag{12}$$

This "global" non-dimensionalization ensures that the SA source term is similar between similar flows, and, additionally, it simplifies the problem as the source is now a function of four quantities, $\hat{\nu}^*, \Omega^*, N^*, s_i{}^*$, rather than five since $\nu = \nu^*$.

It is possible to further non-dimensionalize the features by relevant *local quantities* that are representative of the state of turbulence. From the Buckingham-Pi theorem we know that there must be a function that relates a locally non-dimensional source term to local non-dimensional quantities. We define local scales,

$\nu + \hat{\nu}$ and d, and introduce

$$\bar{\Omega} = \frac{d^2}{\hat{\nu} + \nu}\Omega \ , \tag{13}$$

$$\bar{N} = \frac{d^2}{(\hat{\nu} + \nu)^2}N \ , \tag{14}$$

$$\begin{aligned}\bar{s}_p &= \frac{d^2}{(\hat{\nu} + \nu)^2}s_p \\ &= c_{b1}(1 - f_{t2})\left(\frac{\chi}{\chi + 1}\right)\left(\bar{\Omega} + \frac{1}{\kappa^2}\frac{\chi}{\chi + 1}f_{t2}\right) \ ,\end{aligned} \tag{15}$$

$$\begin{aligned}\bar{s}_d &= \frac{d^2}{(\hat{\nu} + \nu)^2}s_d \\ &= \left(\frac{\chi}{\chi + 1}\right)^2 c_{w1}f_w \ ,\end{aligned} \tag{16}$$

$$\begin{aligned}\bar{s}_{cp} &= \frac{d^2}{(\hat{\nu} + \nu)^2}s_{cp} \\ &= \frac{c_{b2}}{\sigma}\bar{N} \ ,\end{aligned} \tag{17}$$

as locally non-dimensional transformations of the dimensional quantities. This local non-dimensionalization has both benefits and drawbacks. On the one hand, this has further simplified the problem as $\bar{s}$ is a function of three variables rather than four, $\bar{\Omega}$, χ, and $\bar{N}$, and data points are consistent across a single dataset. However, this non-dimensionalization comes with a cost. The non-dimensionalizers, $d^2/(\nu+\hat{\nu})^2$ and $d/(\nu+\hat{\nu})^2$ have poor numerical properties. As d gets large, for example on the outer boundaries of an airfoil mesh, so do $\bar{\Omega}$, $\bar{N}$, and $\bar{s}$. This poor scaling not only makes it difficult for the learning algorithm to find the (relatively) small window of relevant differences in input values, but it is also easy to have extreme outliers, especially problematic for evaluating unseen data locations (this is not a problem for the original SA model because terms are only divided by d). In order to re-dimensionalize the source term value, it is necessary to multiply by $(\nu + \hat{\nu})^2/d^2$, which is poorly behaved near the wall. It is possible that the values for $\hat{\nu}$ and d could be thresholded to avoid such scaling difficulties, but it is difficult to know what that threshold should be.

2.2.3 Loss function

In machine learning, the loss function serves as the objective function for choosing algorithm parameters. It *defines* the quality of a specific prediction, as well as the relative quality between two different predictions. As discussed in Section 2.2.1, the free parameters of the machine learning algorithm are chosen to minimize the sum of the loss function over all of the training points, and so assuming the training data cannot be fit perfectly (which is almost always the case), the loss function dictates the proper balance of prediction errors. For example, a squared loss function,

$$L = \sum_{i=1}^{k}(p_i - t_i)^2 \ , \tag{18}$$

more heavily penalizes predictions that are far away from the truth, whereas a "Manhattan" loss function,

$$L = \sum_{i=1}^{k}|(p_i - t_i)| \ , \tag{19}$$

penalizes outliers relatively less. A machine learning algorithm trained using a squared loss function will often reduce the error in the most incorrect prediction at the expense of being slightly incorrect at many

other locations, whereas an algorithm trained using a manhattan loss function will be more accurate at many points at the cost of having some extreme outliers. The choice of loss function is left up to the user, and should reflect the cost of misprediction inherent in the problem.

When embedded into CFD, the learned algorithm defines a part of the PDEs to be solved. During training, the cost of misprediction should reflect the deviation from the underlying physics that resulting from the ML source term. Unfortunately, such a loss function is difficult to use directly. It is numerically expensive, as it requires all the flows of interest to be recomputed at every iteration of the machine learning training procedure to measure the error caused. Additionally, early iterations of the training procedure will be highly unphysical turbulence models, which will likely cause instabilities in the CFD solver. This would make it impossible to use gradient-based optimizers to find the best set of parameters. It is desirable, therefore, to have a loss function whose measure of quality requires only the training data and not additional PDE solutions. When modeling the locally non-dimensionalized source term, the squared loss function does not appropriately reflect the cost of misprediction. In the PDE solver, fluxes are computed as a function of dimensional source values, i.e. the output of the ML algorithm multiplied by $(\nu + \hat{\nu})^2/d^2$. If this is large, then a small error in the prediction of the non-dimensional source will be amplified into a large error in the dimensional source. One way to fix this discrepancy is to predict the non-dimensional source term, but now to use a squared loss function on the dimensionalized source term

$$L_2 = \sum_{i=1}^{k} \left(\left(\frac{d_i^2}{(\hat{\nu}_i + \nu_i)^2} \right) p_{\bar{s},i} - t_{s,i} \right)^2 . \tag{20}$$

This loss function penalizes differences in the quantity that is relevant to the PDE solver, and thus may be a better measure of the effect on the true system. Note that this loss function is not the same as weighting data points based on the non-dimensional constant, i.e.

$$L_3 = \sum_{i=1}^{k} \frac{d_i^2}{(\hat{\nu}_i + \nu_i)^2} \left(p_{\bar{s},i} - t_{\bar{s},i} \right)^2 . \tag{21}$$

In this latter formulation, if $d_i^2/\hat{\nu}_i + \nu_i)^2$ is close to zero, then every very large differences between the true and predicted values will have a small loss. In the first formulation, the augmented prediction is incentivized to be close to the true value at all data points.

2.3 Results

We have exercised this methodology to learn and replace the Spalart-Allmaras turbulence model. We explored the behavior with a variety of in input and output features, loss functions, and training and testing datasets.

In the results below, we considered at three main sets of training flow solutions. All flows were computed using the using the Stanford University Unstructured [43] flow solver, a median-dual vertex-based code with an inflow Mach number of 0.2. The first set is the development of a turbulent boundary layer over a flat plate with zero pressure gradient, in which the flow was computed at Reynolds numbers of 3, 4, 5, 6 and 7 million. The second set is the development of turbulence during pressure driven flow through a channel at a Reynolds number of 5 million where the outlet pressure is taken as $p_{out} = p_{in} + c_p * \rho * u_{inf}^2$. The flow was computed at $c_p = \{-0, 3, -.1, -0.03, -0.01, 0.01, 0.03, 0.1, 0.3\}$. Both the flat plate and channel flows were computed on a cartesian mesh which has 137 cells in the x-direction and 97 cells in the y-direction. It was run with a farfield boundary condition on the top surface for the flat plate flows and a symmetry boundary condition for the channel flows. The third set of cases was the NACA 0012 airfoil at a Reynolds number of 5 million with an angle of attack sweep between 0 and 12 degrees. These flows were used as three different training data sets:

1. Flat plate at Re = 3, 5, and 7 million.

2. All of the pressure-driven channel flows.

3. All flat plate, channel, and airfoil flows.

For each learning case, every flow was used for testing. There is one datapoint for each grid location in the boundary layer of the flow, which is defined as defined as the locations where $\sqrt{U_i U_i} < U_{inf}$ and $f_{wake} < 0.5$. The definition for f_{wake} is:

$$S_{ij} = 0.5\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \tag{22}$$

$$S_m = \sqrt{2 S_{ij} S_{ij}} \tag{23}$$

$$R_s = \rho * S_m * d * d/(0.09\mu) \tag{24}$$

$$f_{wake} = exp(-1^{-10} R_s R_s) \ . \tag{25}$$

This term distinguishes the wake of the airfoil from the boundary layer.

In each case, learning was conducted using a feed-forward neural network with two hidden layers each containing fifty neurons. At the start of the learning procedure, the data points were linearly scaled such that each input and output has a mean of zero and a variance of one over the whole data set. In all cases, the parameters of the net were trained using the BFGS gradient-based optimizer which is stopped when either the average loss function value is below 10^{-6} or 10,000 function evaluations have occurred. Once training was completed, the RANS solver was modified to call the neural network instead of the standard SA model for the boundary layer points. The flow was then re-run starting from uniform flow with the new turbulence model.

2.3.1 Loss Function comparison

We explored the differences between the squared-distance loss function and dimensionalized loss function discussed previously. The same experimental set-up was used as above, in which training data was taken from the flat plate solutions at 3, 5, and 7 million Reynolds number, and was additionally tested on 4 and 6 million. We tested the two different loss functions for learning the non-dimensional source term $\bar{s}$. Figures 1, 2, and 3 show the results from the squared-distance loss function. Figures 4and 5 show the results from the dimensionalized loss function. It is clear that the dimensionalized loss function greatly outperforms the standard loss function in this case. The squared-distance loss function only penalizes differences in $\bar{s}$, and Fig. 2 shows that the neural net is predicting $\bar{s}$ quite well. However, as seen in Fig. 3, these differences are magnified when they are redimensionalized, creating large differences in the actual value of the source term. In contrast, when the neural net is penalized for dimensional differences, the source term is well-replicated as is the mean flow.

2.3.2 Extended exploration

We examined replacing a number of different parts of the boundary layer:

1. $\bar{s}_d = f(\bar{\Omega}, \chi)$

2. $f_w = f(\bar{\Omega}, \chi)$

3. $m_d = f(\bar{\Omega}, \chi)$

4. $m_p = f(\bar{\Omega}, \chi)$

5. $\bar{s}_p = f(\bar{\Omega}, \chi)$

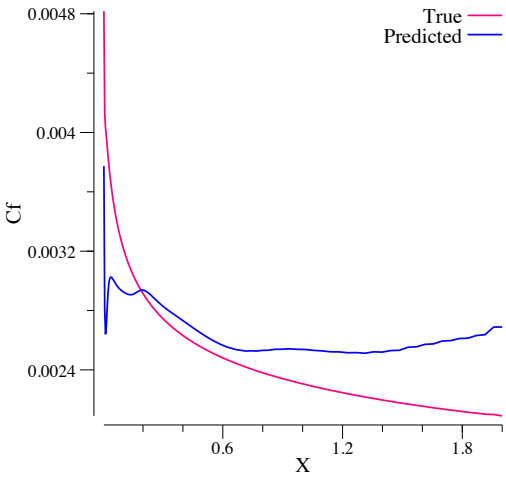


Figure 1: Comparison of skin friction coefficient learning the full non-dimensional source term with the squared loss function.

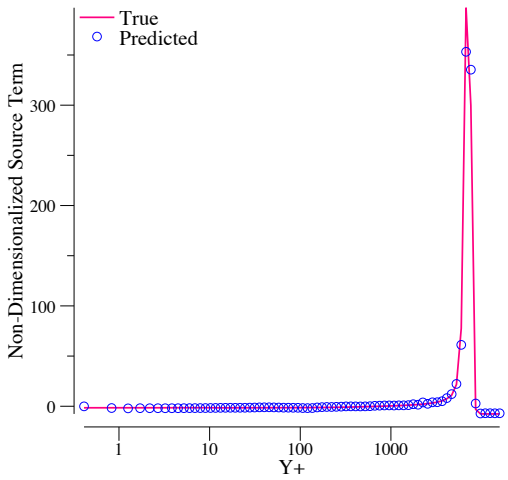


Figure 2: Comparison of the learned non-dimensional source term with the squared loss function.

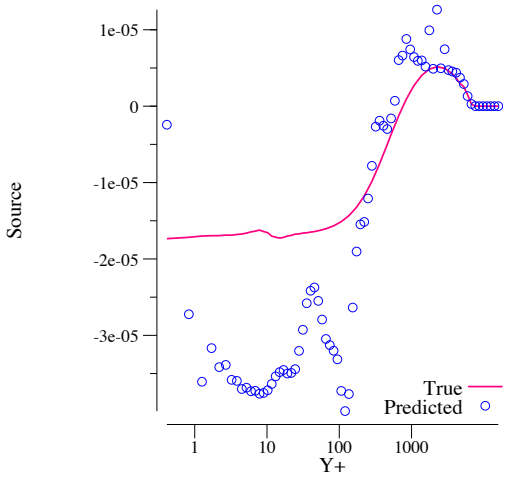


Figure 3: Comparison of the dimensional source term while learning the non-dimensional source term with the squared loss function.

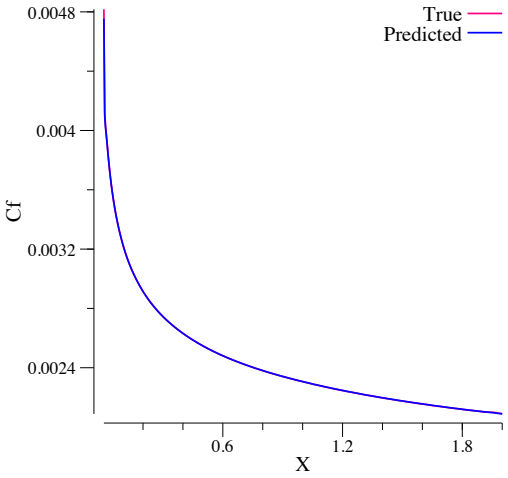


Figure 4: Comparison of skin friction coefficient with the full dimensional source term learned.

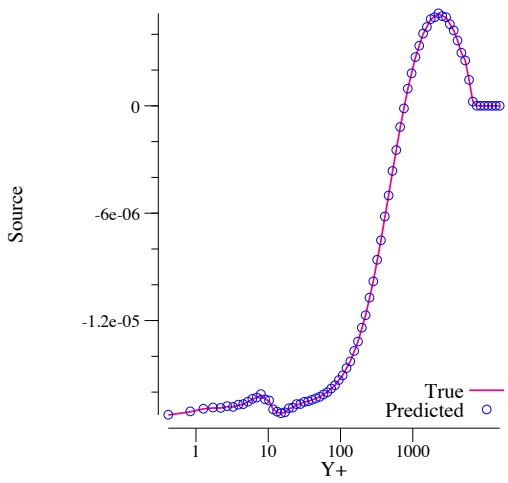


Figure 5: Comparison of the learned dimensional source term at x=2.

6. $\bar{s} = f(\bar{\Omega}, \chi, \bar{N})$

For the non-dimensional quantities, f_w, m_d, and m_p, a squared-difference loss function was used, and for the dimensional quantities, $\bar{s}_d$, $\bar{s}_p$, and $\bar{s}$, the dimensionalized loss function (as discussed in 2.2.2.2.3) was used.

The results can be seen in tables 1, 2, and 3. A bold table entry denotes that the solution was part of the training data. All non-bold table entries are evaluations of the machine-learned turbulence model on unseen data. The table compares the resulting c_f after the flow recomputation with the original c_f from the flow. A perfect machine learned model would reproduce the behavior of SA and would show no differences with the original model. The three categories are:

1. P = Poor, signifying large-scale difference (Fig. 6 , Fig, 7).
2. F = Fair, signifying a significant difference in a small portion of the domain (Fig. 8 , Fig, 9).
3. G = Good, signifying an identical or nearly identical flow solution (Fig. 10 , Fig, 11).

In general, we see excellent agreement for a wide variety of training data sets and source term replacements. The majority of the trials yield positive results, and are able to reproduce the correct flow solution with high accuracy. In particular, Tab. 1 shows the ability of ML to project to new flows. A model trained on flat plate boundary layer data correctly predicts c_f for a NACA 0012 airfoil. Additionally, neural networks appear to be stable when embedded within a PDE solver, as the solver did not diverge in any of the conditions, even those with a poorly-predicting model. These cases were run without tweaks to the configuration of the flow solver to improve convergence (CFL number, multi-grid settings, etc.). In general, these results highlight the ability of ML to be used to augment a turbulence model.

These results also highlight some of the difficulties with the ML process. ML can perform poorly when projecting to new data scenarios, as seen by the relatively poor performance on the channel flows when learning the source from only flat plate solutions (Tab. 1). When a more relevant dataset is used to learn the source term (Tab. 2 and 3), the solution is correctly reproduced.

It is clear that replacing m_p is less robust than the other terms. The reason for this is due to the numerical properties of m_p, specifically the $\hat{\nu}/d^2$ term in the definition of $\hat{S}$. This creates extreme outliers in the ML

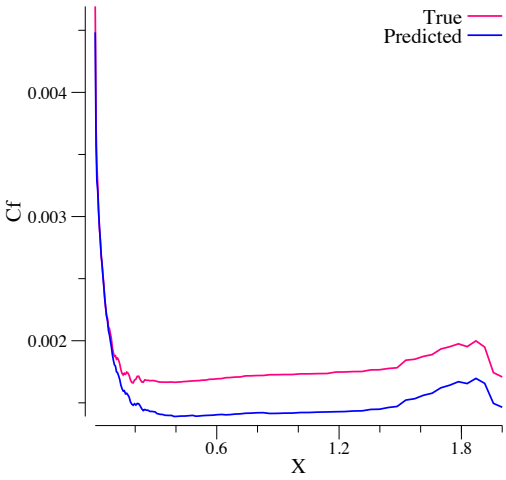


Figure 6: Example of a comparison marked as Poor. This is replacing $\bar{s}_d$ in the boundary layer of the $c_p = 0.3$ channel using training data the three flat plate flows.

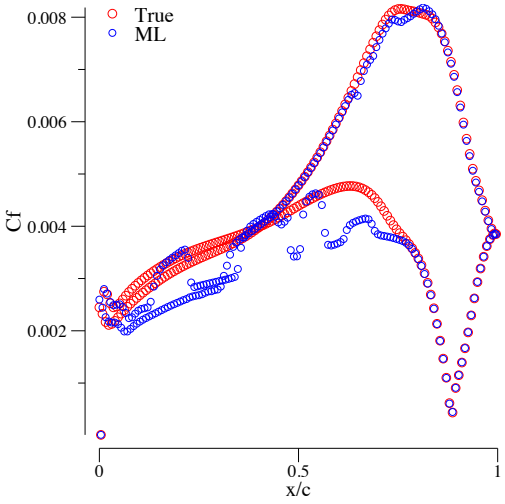


Figure 7: Example of a comparison marked as Poor. This is replacing m_p in the boundary layer of the NACA 0012 airfoil at 2 °using training data from the pressure driven channels.

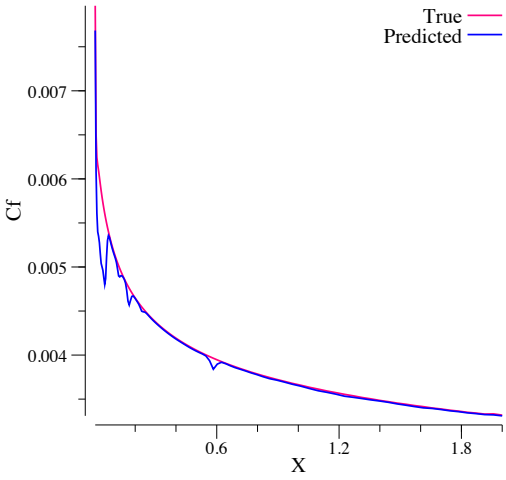


Figure 8: Example of a comparison marked as Fair. This is replacing m_p in the boundary layer of the flat plate at $Re = 4 * 10^6$ using training data the pressure driven channels.

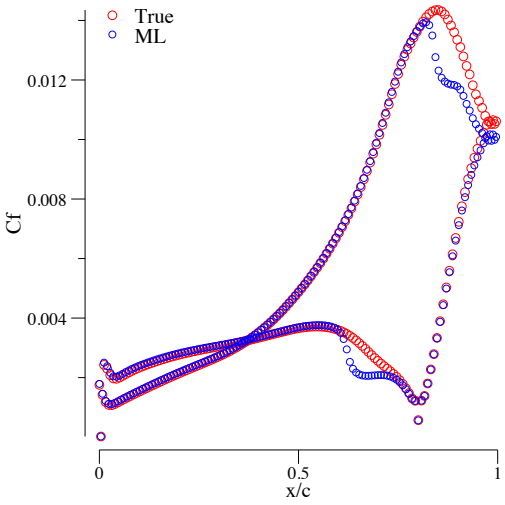


Figure 9: Example of a comparison marked as Fair. This is replacing $\bar{s}$ in the boundary layer of the NACA 0012 airfoil at 5 °using training data from the pressure driven channels.

	Dest.	F_w	Mul. Dest.	Mul. Prod.	Prod.	Source
Flatplate 3e6	G	G	G	G	G	G
Flatplate 4e6	G	G	G	G	G	G
Flatplate 5e6	G	G	G	G	G	G
Flatplate 6e6	G	G	G	G	G	G
Flatplate 7e6	G	G	G	G	G	G
Channel $C_p = -0.3$	G	G	G	G	G	F
Channel $C_p = -0.1$	G	G	G	G	G	F
Channel $C_p = -0.03$	G	G	G	G	G	F
Channel $C_p = -0.01$	G	G	G	G	G	F
Channel $C_p = 0.01$	G	G	G	G	G	F
Channel $C_p = 0.03$	G	G	G	G	G	F
Channel $C_p = 0.1$	G	G	G	G	G	F
Channel $C_p = 0.3$	P	G	G	G	P	F
NACA 0°	G	G	G	G	G	G
NACA 1°	G	G	G	G	G	G
NACA 2°	G	G	G	G	G	G
NACA 3°	G	G	G	G	G	G
NACA 4°	G	G	G	G	G	G
NACA 5°	G	G	G	G	G	G
NACA 6°	G	G	G	G	G	G
NACA 7°	G	G	G	G	G	G
NACA 8°	G	G	G	F	G	G
NACA 9°	G	G	G	F	G	G
NACA 10°	G	G	G	F	G	G
NACA 11°	G	G	G	F	G	G
NACA 12°	G	G	G	F	G	G

Table 1: Results from replacing parts of the SA source term with a neural network using training data from flat plates at $Re = 3, 5, 7$ million. The flow was re-initialized from uniform flow, and the letters represent how well the newly converged flow solution matches the original SA flow solution.

	Dest.	F_w	Mul. Dest.	Mul. Prod.	Prod.	Source
Flatplate 3e6	G	G	G	F	G	P
Flatplate 4e6	G	G	G	F	G	F
Flatplate 5e6	G	G	G	F	G	F
Flatplate 6e6	G	G	G	F	G	F
Flatplate 7e6	G	G	G	F	G	F
Channel $\mathbf{C_p = -0.3}$	G	G	G	P	G	G
Channel $\mathbf{C_p = -0.1}$	G	G	G	P	G	G
Channel $\mathbf{C_p = -0.03}$	G	G	G	P	G	G
Channel $\mathbf{C_p = -0.01}$	G	G	G	P	G	G
Channel $\mathbf{C_p = 0.01}$	G	G	G	P	G	G
Channel $\mathbf{C_p = 0.03}$	G	G	G	P	G	G
Channel $\mathbf{C_p = 0.1}$	G	G	G	P	G	G
Channel $\mathbf{C_p = 0.3}$	G	G	G	F	G	F
NACA 0°	G	G	G	G	G	F
NACA 1°	G	G	G	P	G	F
NACA 2°	G	G	G	P	G	F
NACA 3°	G	G	G	P	G	F
NACA 4°	G	G	G	G	G	F
NACA 5°	G	G	G	G	G	F
NACA 6°	G	G	G	G	G	F
NACA 7°	G	G	G	G	G	F
NACA 8°	G	G	G	G	G	G
NACA 9°	G	G	G	G	G	G
NACA 10°	G	G	G	G	G	G
NACA 11°	G	G	G	G	G	G
NACA 12°	G	G	G	G	G	G

Table 2: Results from replacing parts of the SA source term with a neural network using training data from the channel flows. The flow was re-initialized from uniform flow, and the letters represent how well the newly converged flow solution matches the original SA flow solution.

	Dest.	F_w	Mul. Dest.	Mul. Prod.	Prod.	Source
Flatplate 3e6	G	G	G	P	G	G
Flatplate 4e6	G	G	G	P	G	G
Flatplate 5e6	G	G	G	P	G	G
Flatplate 6e6	G	G	G	P	G	G
Flatplate 7e6	G	G	G	P	G	G
Channel $\mathbf{C_p = -0.3}$	G	G	G	F	G	G
Channel $\mathbf{C_p = -0.1}$	G	G	G	F	G	G
Channel $\mathbf{C_p = -0.03}$	G	G	G	F	G	G
Channel $\mathbf{C_p = -0.01}$	G	G	G	F	G	G
Channel $\mathbf{C_p = 0.01}$	G	G	G	F	G	G
Channel $\mathbf{C_p = 0.03}$	G	G	G	F	G	G
Channel $\mathbf{C_p = 0.1}$	G	G	G	F	F	G
Channel $\mathbf{C_p = 0.3}$	G	G	G	F	F	G
NACA 0°	G	G	G	F	G	G
NACA 1°	G	G	G	F	G	G
NACA 2°	G	G	G	F	G	G
NACA 3°	G	G	G	F	G	G
NACA 4°	G	G	G	F	G	G
NACA 5°	G	G	G	F	G	G
NACA 6°	G	G	G	F	G	G
NACA 7°	G	G	G	F	G	G
NACA 8°	G	G	G	G	G	G
NACA 9°	G	G	G	G	G	G
NACA 10°	G	G	G	G	G	G
NACA 11°	G	G	G	G	G	G
NACA 12°	G	G	G	G	G	G

Table 3: Results from replacing parts of the SA source term with a neural network using training data from all of the example flows. The flow was re-initialized from uniform flow, and the letters represent how well the newly converged flow solution matches the original SA flow solution.

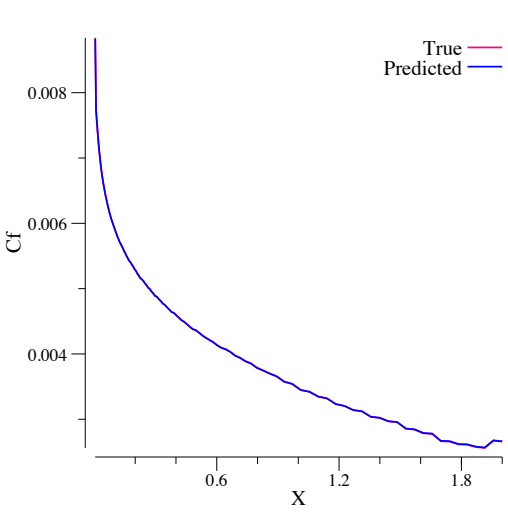


Figure 10: Example of a comparison marked as Good. This is replacing $\bar{s}$ in the boundary layer of the $c_p = -0.3$ channel using training data from all of the flow solutions.

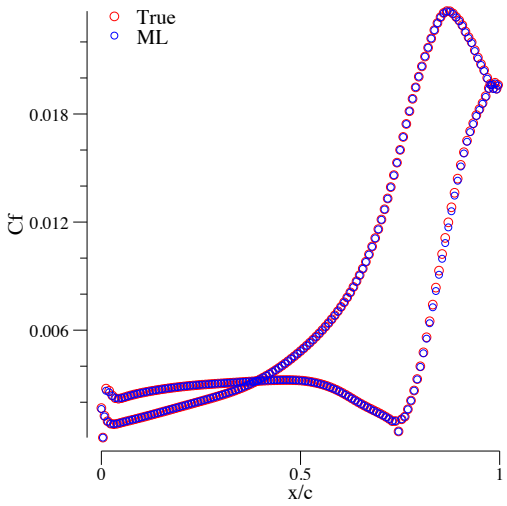


Figure 11: Example of a comparison marked as Good. This is replacing $\bar{s}$ in the boundary layer of the NACA 0012 airfoil at 8°using training data from the three flat plate solutions.

input data that cause difficulties in the learning procedure. For example, the range of m_p in the flat plate solutions is between 0.3 and 1.4 with a reasonable density of data points along the entire range. However, there are a small number of data points in the channel flows where $m_p < -500$, and there are a small number of locations in the NACA 0012 solutions with a value of $m_p < -10000$. In contrast, f_w has limiters built-in to the model, and has no values less than 0 or greater than 2 in any of the flows. We see that the learning of f_w is notably more robust, and performs well across all datasets as a result. This is in spite of the fact that f_w is a more complicated function to reproduce. This highlights the importance of data treatment; it is important to choose input and output features that avoid extreme outliers.

An additional important finding is that the accuracy of the ML algorithm at the converged flow state is not necessarily indicative of success when used to converge the flow solution starting from uniform flow. As an example, Fig. 12 shows the ML prediction for SA source at the converged flow state for the channel flow with $c_p = -0.3$. The agreement looks very good, and yet the eventual flow solution showed significant deviation. Similarly, Fig. 13 shows the prediction of m_p at the converged NACA 0012 flow state at 3 °angle of attack. While the prediction looks quite poor, the eventual flow state shows good agreement. In general, poor ML performance does in fact lead to poor flow solutions. The prediction of m_p is special because at many data points $\Omega\hat{\nu}$ is small, so even large discrepancies in m_p have a negligible impact on the final flow solution (as in Fig. 13). We find that good ML performance is, in general, a necessary, but not sufficient, condition for good online performance. There are many opportunities to deviate from the correct solution, especially in a non-linear system like the RANS equations. The ML algorithm is trained on only converged flow solutions, and flow states at non-converged conditions, such as will occur during the flow solution, can look quite different from the final state. Furthermore, minor errors in the prediction can lead the solution further astray as the solver feeds those errors back upon themselves in the iterative process of convergence.

3 Inverse Modeling

The previous section provided the proof-of-concept that machine learning techniques can replace key terms in a turbulence model, which was assumed to be the surrogate truth. If we are to make advances in turbulence modeling, however, the key step of inference has to be applied to convert data into information that the model

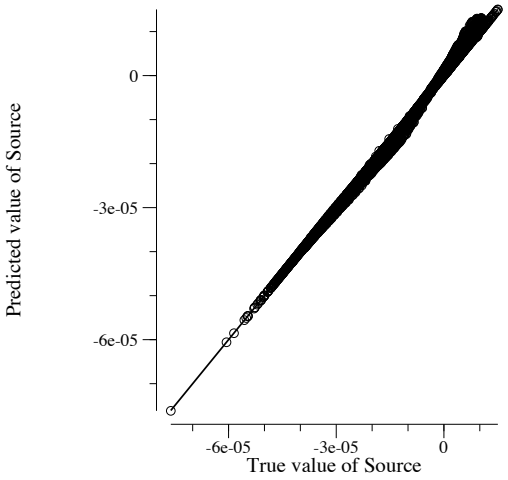


Figure 12: Predicted source value vs. true source value for the channel flow at a c_p of -0.3. The predictions are on the boundary layer points at the converged SA flow state.

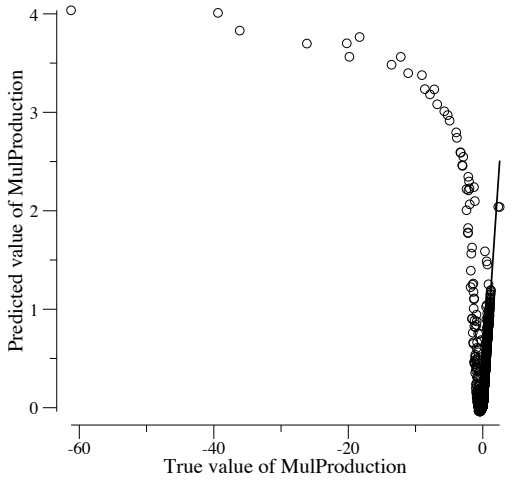


Figure 13: Predicted source value vs. true source value for the NACA 0012 airfoil at 3 $^\circ$angle of attack. The predictions are on the boundary layer points at the converged SA flow state.

is consistent with. For this case, we use inverse modeling. The focus of Phase I is to introduce the general ideas and evaluate the promise of our approach, and hence the demonstrations will be restricted to simple eddy-viscosity-based closures for transition and turbulence.

3.1 Application to a turbulence modeling problem

The closure model that will be considered for the study is the Spalart-Allmaras turbulence model [8]. In this model, a transport equation for a surrogate variable $\tilde{\nu}$ of the turbulent eddy viscosity ν_t is employed. The eddy-viscosity surrogate $\tilde{\nu}$ is transported according to

$$\frac{D\tilde{\nu}}{Dt} = P(\tilde{\nu}) - D(\tilde{\nu}) + T(\tilde{\nu}), \tag{26}$$

where the production P, the destruction D and the diffusion T terms are non-linear terms that are modeled empirically. The above equation is used with a non-linear functional relationship to derive ν_t from $\tilde{\nu}$, which is then used in a Boussinesq formulation to compute the Reynolds stresses. Extracting $\tilde{\nu}$, say from DNS or LES data is of little use without knowing how it will be used in a model. Further, since these equations contain no explicit physical or data parameters, not much benefit can be derived from parameter estimation. The issue at hand, rather, is that **the functional forms of the terms in Eq. 26 are themselves inaccurate**, and in some cases, substantially so.

The chosen application involves the modeling of a non-equilibrium turbulent boundary layer. Specifically, the evolution of a boundary layer over a convex wall is studied, with the computational domain shown in Fig. 14. The geometry corresponds to that used in the experimental measurements of Webster, Degraaf and Eaton[10]. In this problem, the top and bottom boundaries are treated as viscous walls. The left boundary is an equilibrium turbulent boundary layer at a momentum thickness Reynolds number $Re_\theta = 12170$, while the right boundary is a subsonic characteristic outflow. We compare the outputs of our model to the wall-modeled LES (WMLES) computed by Radhakrishnan and Piomelli[11].

As a demonstrative example of the proposed approach, a spatio-temporal function $\alpha(\mathbf{x}, t)$ is introduced

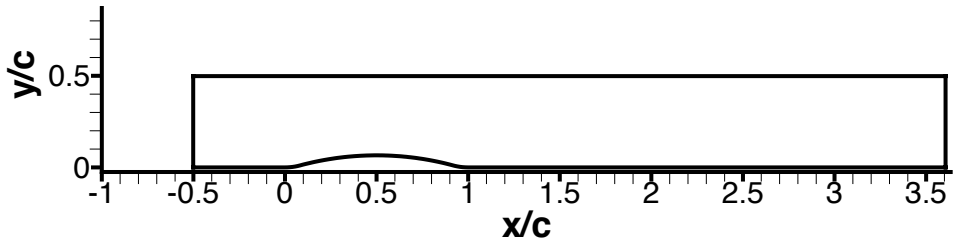


Figure 14: Computational Domain.

into Eq. 26 in the following form:

$$\frac{D\tilde{\nu}}{Dt} = P(\tilde{\nu}) - \alpha(\mathbf{x}, t)D(\tilde{\nu}) + T(\tilde{\nu}). \tag{27}$$

In this paper, we will work with statistically steady problems, so α will be assumed to just have spatial dependence. Assuming we have data $\mathbf{G}_d$, which is a scalar or a vector, a Bayesian[1] (or Least-squares-based frequentist) inverse problem can be constructed to infer $\alpha|G_d$. In this specific example, a classical deterministic optimization with no regularization, *i.e.*

$$\alpha|G_d = arg \ \min_{\alpha} \mathcal{J} = arg \min_{\alpha} ||\mathbf{G}_d - \mathbf{G}_\alpha||_2 \tag{28}$$

s used. Since skin friction data is available from LES, the following objective function was used:

$$\mathcal{J} = \int_w \left[\tau_w^{data}(s) - \tau_w^{model}(s)\right]^2 ds, \tag{29}$$

where the subscript w denotes the lower wall of the computational domain.

The resulting values of $\alpha(\mathbf{x})$ are analogous to the maximum a posteriori (MAP) estimate in a Bayesian inversion with an iteratively updated prior. It may be argued that functional error also exists in the Boussinesq approximation and it could overwhelm the error arising from deficiencies in the transport equations for turbulent scalars. While this assertion may be true in situations such as those with strong secondary flows, an appropriate eddy viscosity (inferred via inverse problems or using a least-squares extraction from high fidelity solutions) was verified to result in a high degree of predictive accuracy of the mean flow quantities in a number of two-dimensional and mildly three dimensional problems that we have investigated. *Thus, before questioning the imperfectness of the Boussinesq approximation, it is important to properly assess the effect of deficiencies* within *eddy-viscosity models.*

In the above approach, α is thus sought at every discrete location in the computational domain, and used in Eq. 27, conjoined with the conservation equations for the ensemble-averaged mass, momentum and energy. The resulting inverse problem is extremely high-dimensional and thus an efficient adjoint-based optimization framework is employed. Details are provided later in this section. This problem is statistically time-independent, and involved inferring α at 14000 spatial locations. Figure 15 compares the baseline solution *i.e.*, with $\alpha(\mathbf{x}) = 1 \ \forall \ \mathbf{x}$ and the solution inferred from the data. Figure 16 shows the distribution of the inferred function α over part of the computational domain. This exercise helps directly quantify modeling inadequacies – **information that is already extremely valuable to the modeler**. Typical practice is to modify one of the above terms using physics-based arguments (for instance, Ref. 9). The inverse modeling procedure gives a quantitative target for functional modification. Alternately, machine learning

[1]It is understood that the use of a Bayesian framework allows for a more rigorous specification of prior information about the model.

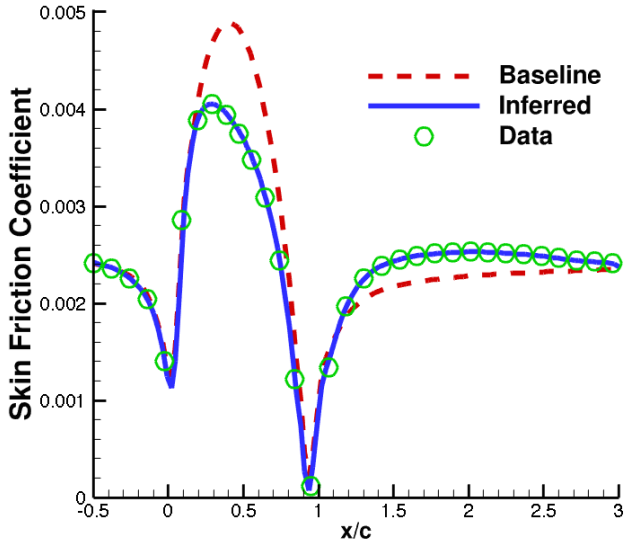


Figure 15: Skin-friction distribution on the lower surface of the computational domain in Figure 14.

methods (introduced in the next section) can also be used. Physically, the higher values of α over the surface of the bump implies a higher rate of destruction of turbulence over the convex surface, which the original model was lacking. It has to be mentioned that, while the quantitative information on model inadequacy is useful, physical interpretations are somewhat loose because all other aspects of the model are also imperfect.

Further, The choice of introducing α as a coefficient function of the destruction term of turbulence model, while being a good physical option, is not necessarily restrictive of the model. Equivalently, a more general function δ could have been introduced as a model discrepancy in the following form

$$\frac{D\tilde{\nu}}{Dt} = P(\tilde{\nu}) - D(\tilde{\nu}) + T(\tilde{\nu}) + \delta. \tag{30}$$

However, both Eq. 27 and 30 are equivalent in that introducing α or δ has fundamentally changed the model, and in fact, if $D \neq 0$, $\delta = (1-\alpha)D$. The advantage of Eq. 27 is that the resulting optimization problem is well-conditioned[2].

3.2 Application to a transition modeling problem

When free-stream turbulence levels are about 1% or more, boundary layers typically proceed from laminar to fully turbulent states without the occurrence of linear instability of the base state: this mode of transition is referred to as bypass transition. Models of bypass transition for general CFD codes are a relatively recent development (Ref. 12) compared to natural transition. At a fundamental level, the bypass process occurs as turbulence diffuses into the laminar boundary layer and generates disturbances known as Klebanoff modes. These grow in amplitude, and transition to turbulence occurs [13]. Closure models, at RANS level, are very loosely based on this mechanism. One method is to use the concept of intermittency γ to blend the flow from the laminar to the turbulent regions. Intermittency is associated with the spottiness of turbulence and manifests itself as a non-Gaussian behavior in turbulent flows. An "intermittency factor" can be formally defined as a fraction of the time turbulence is active, and modeling strategies are roughly based on this definition. *The intermittency factor is not a property of the simulation/experimental database, but is rather a property of the closure model.*

[2]This is because of two reasons: First, α is unit-less and second, the posterior value of α can be expected to be of the same order as its prior value of 1, whereas δ will have to be initiated at 0.

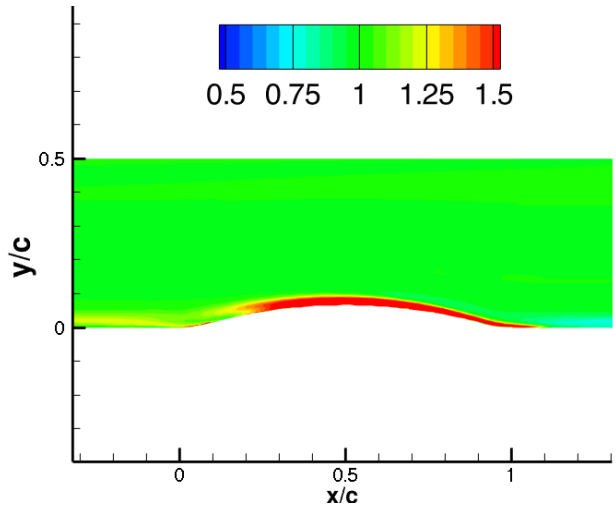


Figure 16: Contour of inferred α

Consider the $k-\omega$ closure (with values of the constants given by Wilcox[17]) and the Reynolds-averaged Navier-Stokes equations. Transition can be introduced by multiplying the production term of the k equation by a function $\gamma(\boldsymbol{x})$. γ is zero in laminar flow, and ramps up to unity in fully turbulent flow. γ appears within the turbulence model only as a factor in the production term of the turbulent kinetic energy transport equation:

$$\frac{Dk}{Dt} = 2\nu_T|S|^2\boldsymbol{\gamma} - C_\mu k\omega + \partial_j\Big[\Big(\nu + \frac{\nu_T}{\sigma_k}\Big)\partial_j k\Big] \tag{31}$$

$$\frac{D\omega}{Dt} = 2C_{\omega 1}|S|^2 - C_{\omega 2}\omega^2 + \partial_j\Big[\Big(\nu + \frac{\nu_T}{\sigma_\omega}\Big)\partial_j \omega\Big] \tag{32}$$

The eddy viscosity ν_T is k/ω.

The model developed by Ge et al.[18] is based on the idea that, in bypass transition under free-stream turbulence, non-zero γ diffuses into the boundary layer, allowing k to be produced, thereby creating eddy viscosity and further enhancing the diffusion of γ. In this way, transition occurs by penetration of free-stream turbulence into the boundary layer via molecular and turbulent diffusion. An intermittency transport equation is defined with a source term, P_γ, that contributes to producing intermittency inside the boundary layers. A sink term, E_γ, ensures that the boundary layer initially is laminar. The form of the model is

$$\frac{D\gamma}{Dt} = \partial_j\Big[\Big(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma}\Big)\partial_j\gamma\Big] + P_\gamma - E_\gamma. \tag{33}$$

A detailed description of the model can be found in Ref. 18.

We begin with the question of determining the intermittency field that will be required within the context of Eq. 32 to match given data on transitional flows. In this problem, the objective function in Eq. 29 is again employed. We will start with a fully turbulent assumption (*i.e.*, $\gamma(\mathbf{x}) = 1$) and attempt to minimize $\mathcal{J}$ by considering every grid point value of γ as parameters in an optimization problem. A sample result of this inverse problem is shown in Figure 17. The output of the problem $\min_\gamma \mathcal{J}$ is thus a data field $\gamma(\mathbf{x})$ that is suited to the $k-\omega$ model. Note that intermittency is defined operationally, in terms of the model and the mechanism of ramping up the production term. It is not a physical variable that can be obtained from data, independently of its use.

Figure 18 shows the result of the inverse solution on the T3-series of test cases[19], which correspond to bypass transition to turbulence over a flat plate with different turbulent intensities and pressure gradients.

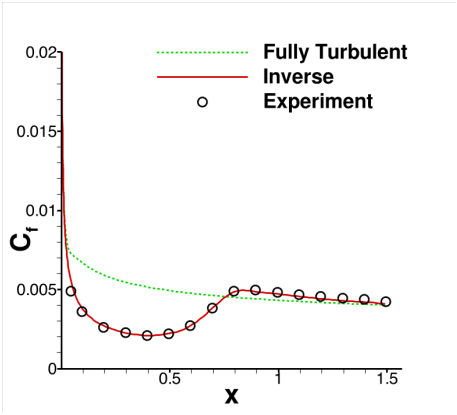
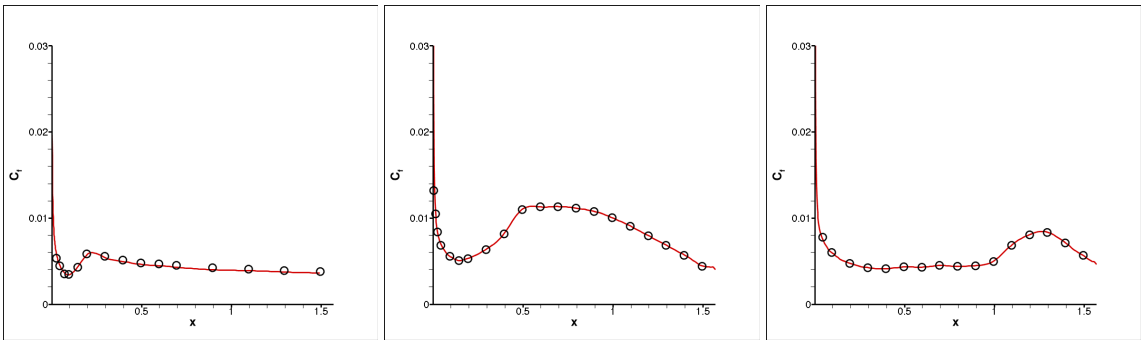


Figure 17: Result of inverse problem to match the experimental skin friction for the T3A test case[19]. The initial condition assumes fully turbulent flow.



(a) T3B (no pressure gradient, $Tu_\infty = 6.5\%$)

(b) T3C1 (pressure gradient, $Tu_\infty = 10\%$)

(c) T3C2 (pressure gradient, $Tu_\infty = 3.7\%$)

Figure 18: Result of inverse solution to match skin friction for T3-series of test cases. Symbols: data; lines: computation.

The left column in Figure 19 shows the inferred intermittency field. The right column shows the intermittency field obtained using the transition model of Ge et al. [18] with the mean flow imposed from the inferred solution. The most significant difference in the T3A and T3B cases is that the inferred field extends higher in the boundary layer near the inlet. This difference explains the the over prediction of C_f near the inlet in the original Ge et al. [18] model and suggests that the sink term in Eq. 33 needs improvement.

The T3C1 case has a high level of free-stream turbulence and shows a prompt transition. Again the inferred field shows the low intermittency extending higher in the boundary layer, near the entrance, than the model. The T3C2 case has a lower free-stream intensity and lower Reynolds number than T3C1 and the C_f prediction in Ref. 18 is fairly accurate; correspondingly, the inferred and modeled fields were confirmed to be fairly close. The cases presented here are representative of the other T3 cases: The inferred intermittency field shows the region of $\gamma < 1$ extending higher into the boundary layer near the inlet and the $\gamma = 1$ region is achieved farther inside the downstream boundary layer. The model postulates a sink term that is a function of $R_t \equiv \nu_T/\nu$ and $R_\nu \equiv d^2|\Omega|/2.188\nu$. It is not clear that the discrepancy between the model and inferred fields can be parametrized by these terms. In the next section, a new parametrization, based on machine learning is proposed to improve the model.

Note that, unlike the turbulence modeling example in which α was constructed as an unknown functional correction, the inference of γ in the transition modeling example is in a different context.

3.3 Optimization procedure

The optimization problem uses a gradient-based Quasi-Newton method employing the limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm [16]. Since the optimization problem is extremely high dimensional (as the number of parameters equals the number of mesh points), an adjoint solver is required to efficiently compute gradients. Consider the discretized governing equations (including boundary conditions) $\mathbf{R}_H(\mathbf{U}_H, \alpha) = \mathbf{0}$, where $\mathbf{U}_H$ represent the conserved variables in the RANS equations along with the turbulent scalars. For a discrete objective function $\mathcal{J}_H$, the discrete adjoint equation[14] for the vector of adjoint variables $\mathbf{\Psi}_H$ is given by

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right]^T \mathbf{\Psi}_H = -\left[\frac{\partial \mathcal{J}_H}{\partial \mathbf{U}_H}\right]^T. \tag{34}$$

In this work, the software suite ADOL-C [15] has been used for automatic differentiation of the complete set of dependencies including the scalar transport variables. Given the adjoint solution, the gradient of the cost function with respect to the intermittency α_i at every mesh point can computed as

$$\frac{d\mathcal{J}_H}{d\alpha_i} = \mathbf{\Psi}_H^T \frac{\partial \mathbf{R}_H}{\partial \alpha_i}$$

and used in the optimization loop.

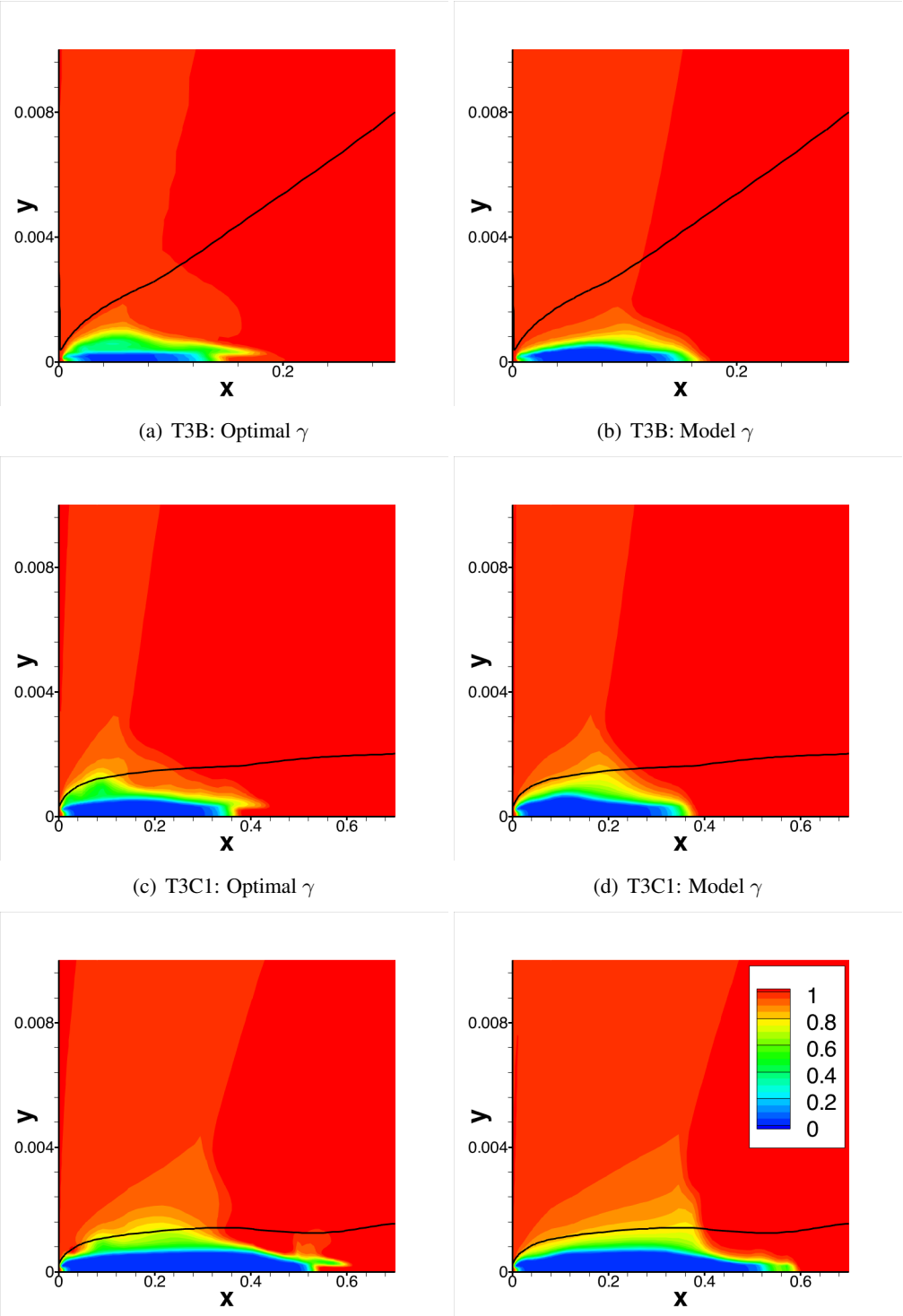
4 Machine Learning Results

In the transition modeling problem, we will first write Eq. 33 as

$$\frac{D\gamma}{Dt} = \partial_j\Big[\Big(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma}\Big)\partial_j\gamma\Big] + S_\gamma. \tag{35}$$

Since convection and diffusion are fundamental transport properties, the functional form of the source term S_γ (*i.e.* production minus destruction) will be targeted for improvement. To be consistent with predictive RANS modeling, S_γ will have to be extracted from the **inferred flow-field** (rather than the DNS or LES) which was computed in the previous section. This can be determined by considering Eq. 35 and computing S_γ by using the following balance (on the optimal mean and intermittency flow-fields)

$$S_\gamma = \frac{D\gamma}{Dt} - \partial_j\Big[\Big(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma}\Big)\partial_j\gamma\Big] \tag{36}$$



(a) T3B: Optimal γ

(b) T3B: Model γ

(c) T3C1: Optimal γ

(d) T3C1: Model γ

(e) T3C5: Optimal γ

(f) T3C5: Model γ

Figure 19: Comparison of inferred intermittency field and the model of Ge et al. [18] for selected cases, with the mean flow imposed from the inferred solution. The line is an iso-contour of 99% of the inlet free-stream velocity.

For the T3-series of test cases, the selected features are

$$\mathbf{q} = \left[k, \omega, \gamma, \frac{\partial u}{\partial y} \; ; \; \frac{\partial v}{\partial y} \; ; d^2\Omega/\nu \right], \tag{37}$$

where Ω is the vorticity magnitude and d is the distance from the nearest viscous wall. These features were selected from a set that included the full velocity gradient tensor, the transported scalars k, ω, γ, and three non-dimensional parameters $\{\nu_t/\nu; d^2\Omega/\nu; \Omega/\omega\}$ that appear in the original Ge et al.[18] model. A standard hill-climbing[21] algorithm was used to narrow down the feature set. Only points within the boundary layer were considered for Machine Learning. Outside the boundary layer, the analytical source term from the baseline model was used. The RANS output was divided into two parts: 80% of the data was used training and 20% for validation. The validation sets were used to adjust hyper-parameters in both Neural Networks (NN) and Gaussian Process (GP) models. Figure 20 plots the original versus the predicted values of S_γ produced by the optimal NN and GP. The SSE for the GP method was found to be four times smaller than that of the NN. It has to be mentioned that the GP method is based on an in-house code for regression and optimization, whereas the NN code uses the FANN[7] library. Though this behavior is representative of the T3-series, this demonstration is preliminary in nature and conclusions about the merits of each ML method cannot be drawn based on the limited number of evaluations.

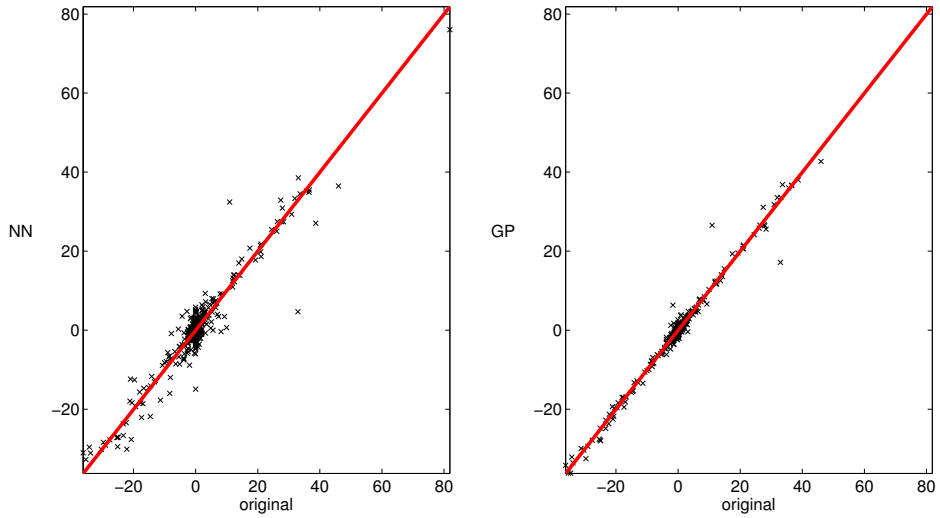
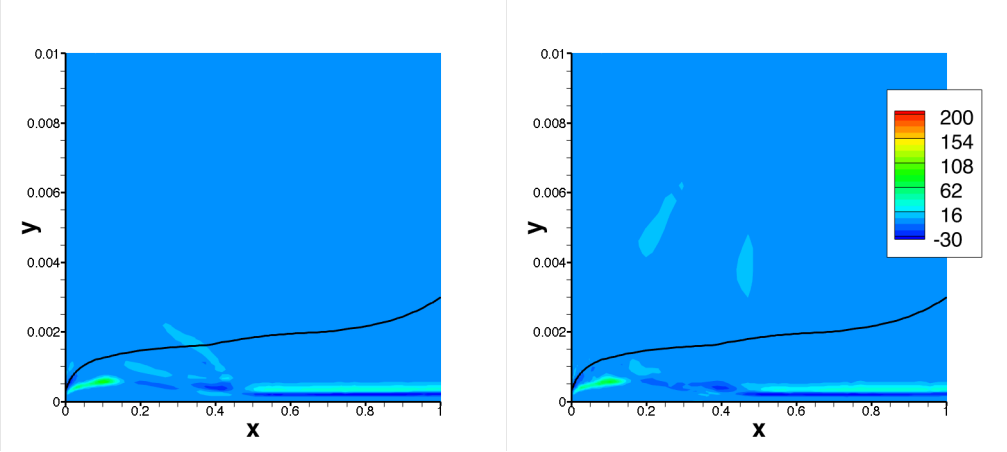


Figure 20: Original versus predicted S_γ for various ML methods for T3C1 transition case.

5 Data Injection Results

The inversion and machine learning steps should be considered as *pre-processing* or *off-line* steps in the proposed framework. During the training stage of the ML model, theoretical knowledge and intuition should be used to inform the ML step of asymptotic behaviors and in regions of sparsely-populated feature space. An example in turbulence modeling would be to use rapid distortion theory to supplement the data set in regions of high deformation rates. During the predictive simulation, (at each time-step or solver iteration), the solver will pass feature vectors $\mathbf{q}_*$ to the ML 'testing' routine and receive appropriate model correction quantities $\boldsymbol{\alpha}_* \approx \mathbf{y}_*(\mathbf{q}_*)$ for injection into the data-driven turbulence model. These quantities are requested at every spatial grid point in the computational domain. If the GP model is used, the information on the probabilistic structure of $\boldsymbol{\alpha}_*$ can be utilized to generate a number of realizations. The **ensemble of simulations can account for the impact of structural uncertainties in the turbulence model**. If an ensemble of simulations are generated, the inevitably large dimension of $\boldsymbol{\alpha}_*$ will require efficient reduced-order spatial and stochastic representations[24, 25] of $\boldsymbol{\alpha}_*$ to keep the computation tractable.



(a) Inferred source term (b) Predicted source term

Figure 21: Comparison of inferred source term and neural network prediction for T3C1.

Simulations of bypass transition, in which the inferred and reconstructed quantity is S_γ in the transition model introduced in the previous section is used as a demonstrative example. Figure 21 compares the predicted source term $S_\gamma(q)$ versus the actual source term $S_\gamma(\mathbf{x})$ via the inverse solution for the T3C1 transition case. The close agreement confirms the validity of the new parametrization. The inferred, predicted and baseline model [18] intermittency profiles for the T3C1 case are shown in Figure 22. For the data injection, initial computations were performed with the baseline model. The machine-learned source term S_γ was then blended into the solution in the form $S_\gamma = (1-\beta)\,[S_\gamma]_{baseline} + \beta\,[S_\gamma]_{ML}$, gradually increasing β from 0 to 1. The inadequacy of the baseline model in predicting the high levels of intermittency required in the context of the $k-\omega$ closure is again confirmed. Note that, to attain the intensity of turbulence in the fully developed region, the intermittency variable has to assume a value greater than unity. This is a failure of the closure model to represent the physics in sufficient detail and thus the role of the intermittency as a model variable, rather than a physical one is emphasized. Figure 23 shows the ability of the machine learning method to reproduce the inferred skin friction results. The noisy nature of the prediction is partly a result of the lack of detail in the data (a total of six T3-series cases were used for the inference). Improvements can be expected by using a larger set of inverse problems as well as by performing the inference with respect to a wider set of objective functions.

6 Summary

The traditional approach to closure modeling does not leverage the availability of large amounts of data from high fidelity simulations and high resolution experiments. The proposed set of approaches highlight the potential of inverse modeling and machine learning techniques to quantify and account for deficiencies in turbulence and transition modeling using data from simulations and measurements. The focus on the functional forms of the closure (rather than on parameters in the model) offers the promise of improved predictive models under the premise that the data is diverse enough to characterize the physical phenomena to be modeled. The inverse modeling step, by itself, provides direct information on the model inadequacy which is of value to the modeler. It has to be mentioned that machine learning should be considered as just one tool to convert the inferred information into modeling knowledge. The modeler can, in principle, use information from the inverse problem to make parametric corrections to existing models. The proposed techniques are general enough to be applied in any physical modeling situation in which appropriate data is available.

In Phase II, work is aimed at inferring model inadequacies in a wider range of problems to specifically

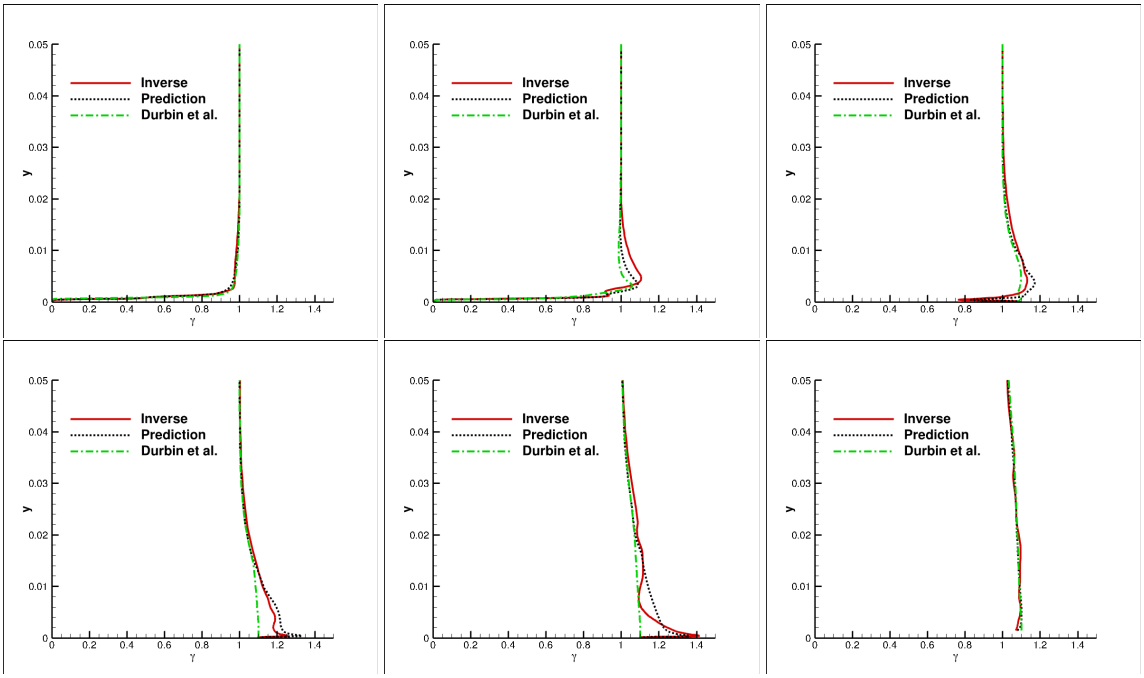


Figure 22: Intermittency field at selected streamwise stations for inferred, predicted and model for T3C1. Locations are x=0.1,0.2,0.4,0.6,1.0,1.5.

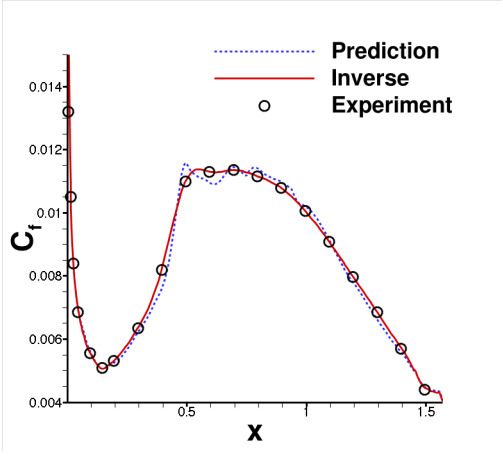


Figure 23: Skin friction prediction for T3C1.

target turbulent flow separation and bypass transition. While Phase I suggests that there is merit in exploring data-driven techniques, several details are being addressed currently. Towards this end, a consistent Bayesian framework for inversion/machine learning/uncertainty propagation is being built, such that the targeted model can offer improved predictions as well as confidence intervals on the predicted outputs.

7 References

[1] G. Gerolymos, E. Sauret & I. Vallet, "Contribution to single-point closure Reynolds-stress modelling of inhomogeneous flow," *Theoretical Computational Fluid Dynamics,* Vol. 17, pp. 407–431, 2004.

[2] B. Younis, T. Gatski & C. Speziale, "Towards a rational model for the triple velocity correlations of turbulence," *Proc. R. Soc. London A*, Vol. 456, pp. 909–920, 2000.

[3] M. Milano & P. Koumoutsakos, "Neural network modeling for near wall turbulent flow," *Journal of Computational Physics,* Vol. 182(1), pp. 1–26, 2002.

[4] I. Marusic, V. Interrante, P. K. Subbareddy & A. Moss, "Chapter 13 - Real-time feature extraction for the analysis of turbulent flows," *Data Mining for Scientific and Engineering Applications,* 2001.

[5] S. Yarlanki, B. Rajendran & H. Hamann, "Estimation of turbulence closure coefficients for data centers using machine learning algorithms," *13^{th} IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems,* San Diego, CA, 2012.

[6] Edeling, W., Cinnella, P., Dwight, R., & Bijl, H., "Bayesian estimates of parameter variability in the $k-\epsilon$ turbulence model," *Journal of Computational Physics*, Vol. 258, pp. 73–94, 2014.

[7] Nissen, S., "Implementation of a fast artificial neural network library (FANN)," *Report, Department of Computer Science University of Copenhagen (DIKU)*, Vol. 31, 2003.

[8] P. R. Spalart & S. R. Allmaras, "A One-Equation Turbulence Model for Aerodynamic Flows," 30^{th} *Aerospace Sciences Meeting & Exhibit*, Reno, NV, Jan 1992.

[9] Shur, M. L., Strelets, M. K., Travin, A. K. & Spalart, P. R., "Turbulence Modeling in Rotating and Curved Channels: Assessing the Spalart-Shur Correction," *AIAA Journal* Vol. 38(5), pp. 784–792, 2000.

[10] Webster, D., DeGraaff, D. & Eaton, J., "Turbulence characteristics of a boundary layer over a two-dimensional bump," *Journal of Fluid Mechanics*, Vol. 320(1), pp. 53–69, 1996.

[11] Radhakrishnan, S., Keating, A., Piomelli, U. & Silva Lopes, A., "Reynolds-averaged and Large-eddy simulations of turbulent non-equilibrium flows," *Journal of Turbulence*, Vol. 7, 2006.

[12] Menter, F. R., Langtry, R. & Volker, S., "Transition modelling for general purpose CFD codes," *Flow, Turbulence and Combustion* Vol. 77(1-4) pp. 277–303, 2006.

[13] Durbin, P. & Wu, X., "Transition beneath vortical disturbances," *Annual Review of Fluid Mechanics*, Vol. 39, pp. 107–128, 2007.

[14] Giles, M. & Pierce, N., "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, Vol. 65(3-4), pp. 393–415, 2000.

[15] Griewank, A., Juedes, D. & Utke, J., "Algorithm 755: ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++," *ACM Transactions on Mathematical Software*, Vol. 22(2), pp. 131–167, 1996.

[16] Dennis, J.E. & More, J., "Quasi-Newton methods, motivation and theory," *SIAM review*, Vol. 19(1), pp. 46–89, 1977.

[17] Wilcox, D., *Turbulence modeling for CFD.,* Vol. 2, DCW industries, 2006.

[18] Ge, X., Arolla, S. & Durbin, P., "A bypass transition model based on the intermittency function," *Flow, Turbulence and Combustion* Vol. 93(1), pp. 37–61, 2014.

[19] Roach, P. E. & Brierley, D. H., "The influence of a turbulent free-stream on zero pressure gradient transitional boundary layer development Part 1: Test cases T3A and T3B," *Numerical Simulation of Unsteady Flows and Transition to Turbulence*, pp. 319–347, ERCOFTAC, 1992.

[20] Bishop, C. M., *Pattern Recognition and Machine Learning,* Springer, 2006.

[21] Kohavi, R. & John, G. H., "Wrappers for Feature Subset Selection," *Artificial Intelligence,* Vol. 97(1) pp. 273–324, 1997.

[22] Park, J. & Sandberg, I. W., "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation,* Vol. 3(2) pp. 246–257, 1992.

[23] Rasmussen, C. E. & Williams, C. K. I., *Gaussian Processes for Machine Learning,* MIT Press, 2006.

[24] Duraisamy, K. & Chandrashekar, P., "Goal oriented uncertainty propagation using stochastic adjoints," *Computers and Fluids* Vol. 66, pp. 10–20, 2012.

[25] Lakshminarayan, V., Duraisamy, K. & Alonso, J., "Estimation and Adaptive Control of Spatial, Temporal and Stochastic Errors Using Adjoints for Unsteady Aerodynamic Applications," 51^{st} *AIAA Aerospace Sciences Meeting,* Grapevine, TX, 2013.

[26] Wilcox, D. C., "Formulation of the kw Turbulence Model Revisited," *AIAA journal*, Vol. 46, No. 11, 2008, pp. 2823–2838.

[27] Menter, F. R., "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA journal*, Vol. 32, No. 8, 1994, pp. 1598–1605.

[28] Slotnick, J. P., Khodadoust, A., Alonso, J. J., Darmofal, D. L., Gropp, W. D., Lurie, E. A., Mavriplis, D. J., and Venkatakrishnan, V., "Enabling the environmentally clean air transportation of the future: a vision of computational fluid dynamics in 2030," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 372, No. 2022, 2014.

[29] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," Technical Publication NASA/CR-2014-218178, NASA, Langley Research Center, Hampton VA 23681-2199, USA, March 2014.

[30] Dow, E. and Wang, Q., "Uncertainty Quantification of Structural Uncertainties in RANS Simulations of Complex Flows," *AIAA Paper*, Vol. 3865, 2011.

[31] Emory, M., Pecnik, R., and Iaccarino, G., "Modeling Structural Uncertainties in Reynolds-Averaged Computations of Shock/Boundary Layer Interactions," *AIAA Paper*, Vol. 479, 2011.

[32] Cheung, S. H., Oliver, T. A., Prudencio, E. E., Prudhomme, S., and Moser, R. D., "Bayesian Uncertainty Analysis with Applications to Turbulence Modeling," *Reliability Engineering and System Safety*, Vol. 96, 2011, pp. 1137–1149.

[33] Oliver, T. and Moser, R., "Bayesian uncertainty quantification applied to RANS turbulence models," *Journal of Physics: Conference Series*, Vol. 318, IOP Publishing, 2011, p. 042032.

[34] Edeling, W., Cinnella, P., Dwight, R. P., and Bijl, H., "Bayesian estimates of parameter variability in the k–ϵ turbulence model," *Journal of Computational Physics*, Vol. 258, 2014, pp. 73–94.

[35] Edeling, W., Cinnella, P., and Dwight, R., "Predictive RANS simulations via Bayesian Model-Scenario Averaging," *Journal of Computational Physics*, Vol. 275, 2014, pp. 65–91.

[36] Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[37] Watson, G. S., "Smooth Regression Analysis," *The Indian Journal of Statistics*, 1964.

[38] Cleveland, W. S., Devlin, S. J., and Grosse, E., "Variance Reduction Methods I," *Monte Carlo Simulation: IEOR E4703*, 2004.

[39] Tracey, B., Duraisamy, K., and Alonso, J. J., "Application of Supervised Learning to Quantify Uncertainties in Turbulence and Combusion Modeling," *41st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition 07-10 January, Dallas Texas*, 2013.

[40] Rosenblatt, F., *The perceptron, a perceiving and recognizing automaton Project Para*, Cornell Aeronautical Laboratory, 1957.

[41] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, 1989, pp. 359–366.

[42] Spalart, P. R., Shur, M. L., Strelets, M. K., and Travin, A. K., "Direct Simulation and RANS Modeling of a Vortex Generator Flow," *10 th Intertanional ERCOFTAC Symposium on Engineering Turbulence Modeling and Measurements, 17-19 Septemeber, Marbella, Spain*, 2014.

[43] Palacios, F., Alonso, J., Duraisamy, K., Colonno, M., Hicken, J., Aranake, A., Campos, A., Copeland, S., Economon, T., Lonkar, A., et al., "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design." *51st AIAA Aerospace Sciences Meeting and Exhibit*, 2013.

[44] Schmitt, V. and Charpin, F., "Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers," *Experimental data base for computer program assessment*, Vol. 4, 1979.